

### Лабораторная работа 3

#### «Переменные, простые типы данных и операции над ними»

**Цель работы:** освоить базовый синтаксис языка Python, простые типы данных, приобрести навыки создания интерактивных программ с использованием линейных алгоритмов.

#### Основное задание

1. Разработать интерактивную программу, демонстрирующую работу с простыми типами данных:

1.1	Программа должна запрашивать имя пользователя (строка) и почтовый индекс (целое число) и записывать соответственно в переменные <i>Chameleon1</i> и <i>Chameleon2</i> . Вывести значения и типы переменных на экран.
1.2	Преобразовать переменную <i>Chameleon2</i> к строковому типу с использованием функции <code>str()</code> . Умножить полученную строку на число даты рождения пользователя и вывести результат.
1.3	Сложить переменные <i>Chameleon1</i> и <i>Chameleon2</i> , результат записать в <i>Chameleon3</i> . Вывести результат суммирования и тип переменной.
1.4	Преобразовать переменную <i>Chameleon3</i> к списку с использованием функции <code>list()</code> . Вывести результат и тип переменной на экран.
1.5	Напечатать начальный и конечный элементы списка <i>Chameleon3</i> . Первое значение списка <i>Chameleon3</i> заменить возрастом пользователя.
1.6	Прибавить к списку <i>Chameleon3</i> список, вводимый пользователем с клавиатуры. Результат вывести на экран, а затем умножить на 2 и снова вывести на экран.
1.7	Преобразовать переменную <i>Chameleon3</i> к кортежу с использованием функции <code>tuple()</code> . Вывести результат и тип переменной на экран.
1.8	Попытаться изменить любое значение кортежа <i>Chameleon3</i> .
1.9	Продемонстрировать операцию сложения кортежей и умножения кортежа на число.
1.10	Преобразовать переменную <i>Chameleon3</i> к множеству с использованием функции <code>set()</code> . Вывести результат и тип переменной на экран. Сделать вывод о назначении множеств.
1.11	Создать словарь <i>My_Dictionary</i> из пяти пар, осуществляющий перевод слов с русского на английский. Продемонстрировать обращение к элементам словаря по ключу. Вывести словарь несколько раз на экран. Сделать вывод.
1.12	Сделать вывод о простых типах данных языка Python, универсальности команд и о динамической типизации.

2. Разработать интерактивную программу «*What is My Age in Seconds*» («Каков мой возраст в секундах»), которая на входе принимает дату рождения пользователя, рассчитывает количество прожитых пользователем секунд и выводит результат на экран монитора.

#### Дополнительное задание\*

Согласно официальному сайту Европейского Центрального Банка в настоящий момент наши западные соседи используют в обращении 7 номиналов банкнот евро – 5, 10, 20, 50, 100, 200 и 500 евро. Монетный ряд включает 8 монет достоинством 1, 2, 5, 10, 20, 50 евроцентов, 1 и 2 евро (1 евро = 100 евроцентов).

Вариант 1. Необходимо написать программу для специального банкомата (АТМ). Банкомат должен оптимальным способом выдавать любую введённую пользователем сумму: вначале выдаются крупные банкноты, потом меньше и так вплоть до одного евроцента. К примеру, пользователь запрашивает в банкомате сумму в 587 евро и 99 центов. Банкомат ему должен выдать следующее: банкноты номиналом 500, 50, 20, 10 и 5 евро, затем монеты: 50,  $20 \times 2$ ,  $5 \times 2$  и 1 евроцент.

Вариант 2. Необходимо написать приложение для разменного автомата. Работа автомата заключается в следующем: автомат принимает у пользователя купюру и выдаёт ему купюры меньшего номинала. Аналогично осуществляется размен монет.

#### Замечания.

1. Студенты, имеющие нечетный номер по списку выполняют первый вариант задания, остальные – второй.

2. Допускается использовать любую другую валюту.

#### Требования к выполнению

1. Программа должна быть обязательно снабжена комментариями, в которых необходимо указать краткое предназначение программы, номер лабораторной работы и её название, версию программы, ФИО разработчика, дату разработки.

2. Каждая программа должна быть снабжена дружелюбным и интуитивно понятным интерфейсом (в консольном варианте).



Перед началом работы необходимо всегда думать о нуждах конечных пользователей, только так можно создать по-настоящему качественный продукт!

### Контрольные вопросы

1. Описать архитектуру и основные элементы компьютера. Какой элемент является центральным при построении любой вычислительной системы?
2. Какие типы памяти доступны при разработке программ?
3. Что такое машинный код?
4. В чём отличия языка высокого уровня от языка низкого уровня?
5. Какой язык понимает и обрабатывает центральный процессор (*Central Process Unit, CPU*)?
6. Что такое транслятор и что он делает?
7. Что общего между компилятором и интерпретатором и чем они отличаются?
8. Что такое переменная и зачем она нужна в программе?
9. Что такое «соглашение по присваиванию имён»?
10. Что такое константа?
11. Что такое *hardcode* («тяжёлый код»)?
12. Описать стандартные простые типы данных в *Python*.
13. Что измеряет диапазон типа данных?
14. Что такое выражение?
15. Что такое инициализация?
16. Можно ли обратиться к переменной, если инициализировать её чуть позже?
17. Что произойдёт, если присвоить строковой переменной целочисленное значение (или наоборот)?
18. Какой из арифметических операторов не может иметь дробного операнда?
19. Какой из арифметических операторов не может иметь в качестве второго операнда значение ноль?
20. Описать все группы операций, доступные в языке *Python*.

## Базовый синтаксис языка *Python*: переменные, простые типы данных и операции над ними

Информация, сохраненная в памяти компьютера, может быть разных типов данных. К стандартным типам данных, используемым в *Python*, относятся: число (*Number*); строка (*String*); список (*List*); кортеж (*Tuple*); словарь (*Dictionary*); множество (*Set*).

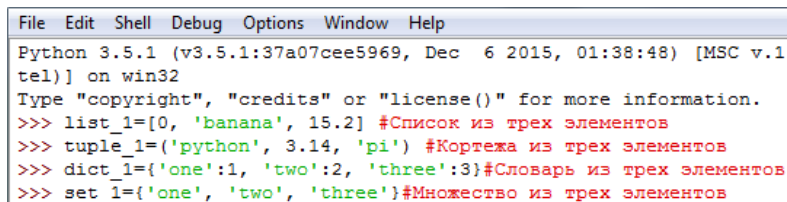
*Числовой тип данных* в *Python* предназначен для хранения числовых значений и представляет собой *неизменяемый тип данных*, то есть изменение значения числового типа приведет к созданию нового объекта в памяти (и удалению старого).

Для хранения и обработки текстовой информации используется *строковый тип данных*. В *Python* строки могут задаваться следующими тремя способами:

- 1) строка в одинарных кавычках (апострофах);
- 2) строка в двойных кавычках;
- 3) строка в тройных кавычках;

Особенность строки в тройных кавычках состоит в том, что она может занимать в коде несколько строк и при этом выводиться на экран точно в таком же виде, как и вводится!

Списки, кортежи, словари и множества – представляют собой последовательности значений произвольных типов (рисунок 3.1).

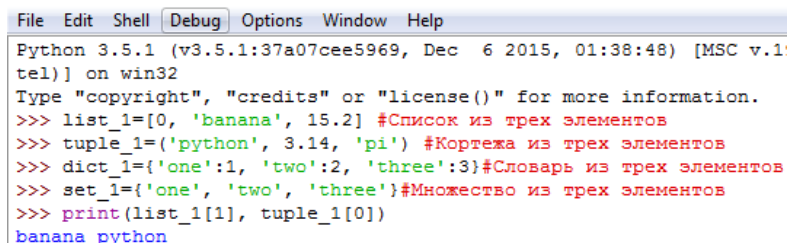


```
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> list_1=[0, 'banana', 15.2] #Список из трех элементов
>>> tuple_1=('python', 3.14, 'pi') #Кортежа из трех элементов
>>> dict_1={'one':1, 'two':2, 'three':3}#Словарь из трех элементов
>>> set_1={'one', 'two', 'three'}#Множество из трех элементов
```

Рис. 3.1. Объявление списка, кортежа, словаря и множества

Числа, строки и кортежи представляют собой *неизменяемые типы данных*, то есть изменение значения переменной каждого типа приведет к созданию нового объекта в памяти (и удалению старого). Списки, множества и словари представляют собой *изменяемые последовательности*.

Обращение к элементам последовательностей осуществляется с помощью квадратных скобок (рисунок 3.2)



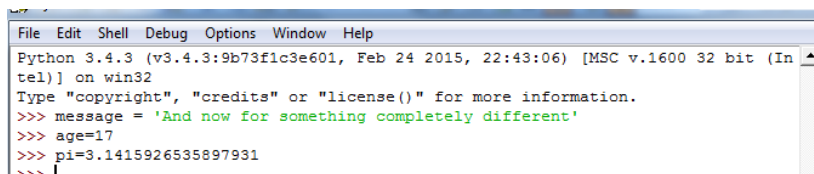
```
File Edit Shell Debug Options Window Help
Python 3.5.1 (v3.5.1:37a07cee5969, Dec 6 2015, 01:38:48) [MSC v.1
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> list_1=[0, 'banana', 15.2] #Список из трех элементов
>>> tuple_1=('python', 3.14, 'pi') #Кортежа из трех элементов
>>> dict_1={'one':1, 'two':2, 'three':3}#Словарь из трех элементов
>>> set_1={'one', 'two', 'three'}#Множество из трех элементов
>>> print(list_1[1], tuple_1[0])
banana python
```

Рис. 3.2. Вывод элементов списка и кортежа

*Переменная* – это имя, которое ссылается на значение.

*Выражение* – комбинация значений, переменных и операторов.

*Оператор присваивания (assignment statement)* « $\leftarrow$ » создает новые переменные и присваивает им значения (рисунок 3.3).



```
File Edit Shell Debug Options Window Help
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> message = 'And now for something completely different'
>>> age=17
>>> pi=3.1415926535897931
>>>
```

Рис. 3.3. Оператор присваивания

```

Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> message = 'And now for something completely different'
>>> age=17
>>> pi=3.1415926535897931
>>> type(message)
<class 'str'>
>>> type(age)
<class 'int'>
>>> type(pi)
<class 'float'>
>>> |

```

Рис. 3.4. Типы переменных

Типы переменных – это типы значений, на которые они ссылаются (рисунок 3.4). Для переменных, как правило, используют имена, которые раскрывают их назначение. Имена переменных должны начинаться с буквы, могут содержать буквы, цифры, символы подчеркивания; нежелательно использовать буквы верхнего регистра и нельзя использовать зарезервированные слова (рисунок 3.5).

```

Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit (In
tel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as', 'assert', 'break', 'class', 'continue', '
def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if',
'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'retu
rn', 'try', 'while', 'with', 'yield']

```

Рис. 3.5. Зарезервированные слова в Python

В Python имеет место *динамическая типизация*, где любой объект является ссылкой, а типом объекта является то, на что он ссылается. Тип может динамически меняться в процессе выполнения программы, когда ссылка начинает указывать на объект другого типа.

## ОПЕРАТОРЫ И ОПЕРАНДЫ

Говоря простым языком, в выражении  $5 + 7$ , числа «5» и «7» называются операндами, знак «+» оператором. В Python существуют следующие *типы операторов*: арифметические операторы, операторы сравнения, операторы присваивания, побитовые операторы, логические операторы, операторы членства и тождественности. В таблицах 3.1 – 3.7 описаны наиболее используемые операторы.

Таблица 3.1 – Арифметические операторы

Арифметические операторы				Описание
+	-	*	/	Сложение, вычитание, умножение и деление.
**				Возведение в степень.
%				Деление по модулю – делит левый операнд на правый и возвращает остаток.
//				Целочисленное деление операндов.

Таблица 3.2 – Строковые операторы

Строковые операторы	Описание
+	Конкатенация ( <i>concatenation</i> ) строки.
*	Дублирование строки несколько раз.

Таблица 3.3 – Строковые операторы

Операторы сравнения	Описание
---------------------	----------

==				Проверка равенства двух операндов.
!=	<>			Проверка неравенства двух операндов. Возвращает ложь, если операнды равны.
>	<	>=	<=	Проверка, больше (меньше, не меньше, не больше) ли левый операнд правого.

Таблица 3.4 – Операторы присваивания

Операторы присваивания	Описание
=	Оператор присваивания.
+=	Присваивание левому операнду значения суммы правого и левого операндов. Например, $x += y$ равносильно: $x = x + y$ .

Таблица 3.5 – Логические операторы

Логические операторы	Описание
and	Логическое «И». Возвращает истину, если оба операнда истинны.
or	Логическое «ИЛИ». Возвращает истину, если хотя бы один из операндов истинный.
not	Логическое «НЕ». Изменяет логическое значение операнда на противоположное.

### ПРИОРИТЕТ ОПЕРАТОРОВ В PYTHON

В таблице 3.6 описан приоритет операторов в *Python*.

Таблица 3.6 – Приоритет операторов в *Python*

Операторы	Описание
**	Возведение в степень
~	Комплиментарный оператор
*, /, %, //	Умножение, деление, деление по модулю, целочисленное деление.
+, -	Сложение, вычитание.
>>, <<	Побитовый сдвиг вправо (влево).
&	Бинарное «И».
^,	Бинарные «Исключительное ИЛИ», «ИЛИ».
>, <, >=, <=	Операторы сравнения.
==, !=, <>	Операторы равенства.
=, +=, -=, ...	Операторы присваивания.
is, is not	Тождественные операторы
in, not in	Операторы членства.
not, or, and	Логические операторы.

### ПРЕОБРАЗОВАНИЕ ТИПОВ ДАННЫХ

Иногда может возникнуть необходимость преобразовать один тип данных в другой. Для этого существуют специальные встроенные функции *Python*. Некоторые из них представлены в таблице 3.7.

Таблица 3.7 – Некоторые встроенные функции в *Python*

Функция	Описание	Пример
int(x [,base])	Преобразование x в целое число.	int(12.4) → 12

<i>long</i> ( <i>x</i> [, <i>base</i> ] )	Преобразование <i>x</i> в тип <i>long</i> .	<i>long</i> (20) → 20 <i>L</i>
<i>float</i> ( <i>x</i> )	Преобразование <i>x</i> в вещественное число.	<i>float</i> (10) → 10.0
<i>str</i> ( <i>x</i> )	Преобразование <i>x</i> в строку.	<i>str</i> (10) → "10"